

图文 91 如何从Github拉取RocketMQ源码以及导入IntelliJ IDEA中?

496 人次阅读 2020-02-06 08:52:59

详情 评论

如何从Github拉取RocketMQ源码以及导入IntelliJ IDEA中?



继《从零开始带你成为JVM实战高手》后，救火队长携新作再度出山，重磅推荐：

(点击下方蓝字试听)

[《从零开始带你成为MySQL实战优化高手》](#)

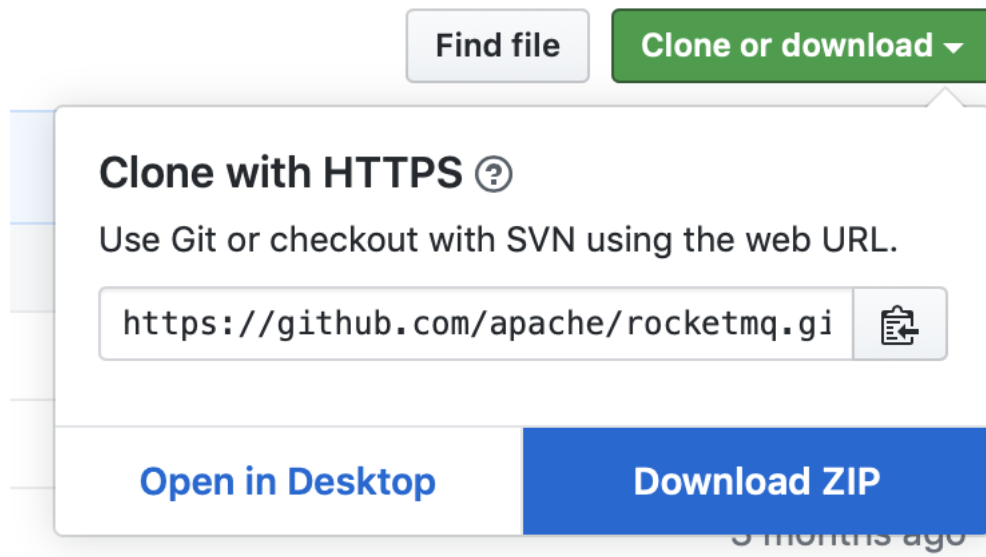
## 1、从Github下载RocketMQ源码

今天开始我们进入本专栏全新增加的一部分内容，就是RocketMQ源码分析，首先我们这一讲会教大家，如果你要分析RocketMQ的源码，应该如何从Github上拉取他的源码下来，然后导入到IntelliJ IDEA里去，同时在本地进行源码的调试。

首先，我建议大家直接在浏览器中进入RocketMQ的github页面，直接在里面下载他的源码到本地，他的地址为：

<https://github.com/apache/rocketmq>

大家进入这个地址之后，可以看到一个“clone or download”的按钮，在里面点击“Download ZIP”可以下载RocketMQ的源码，我们看下图。

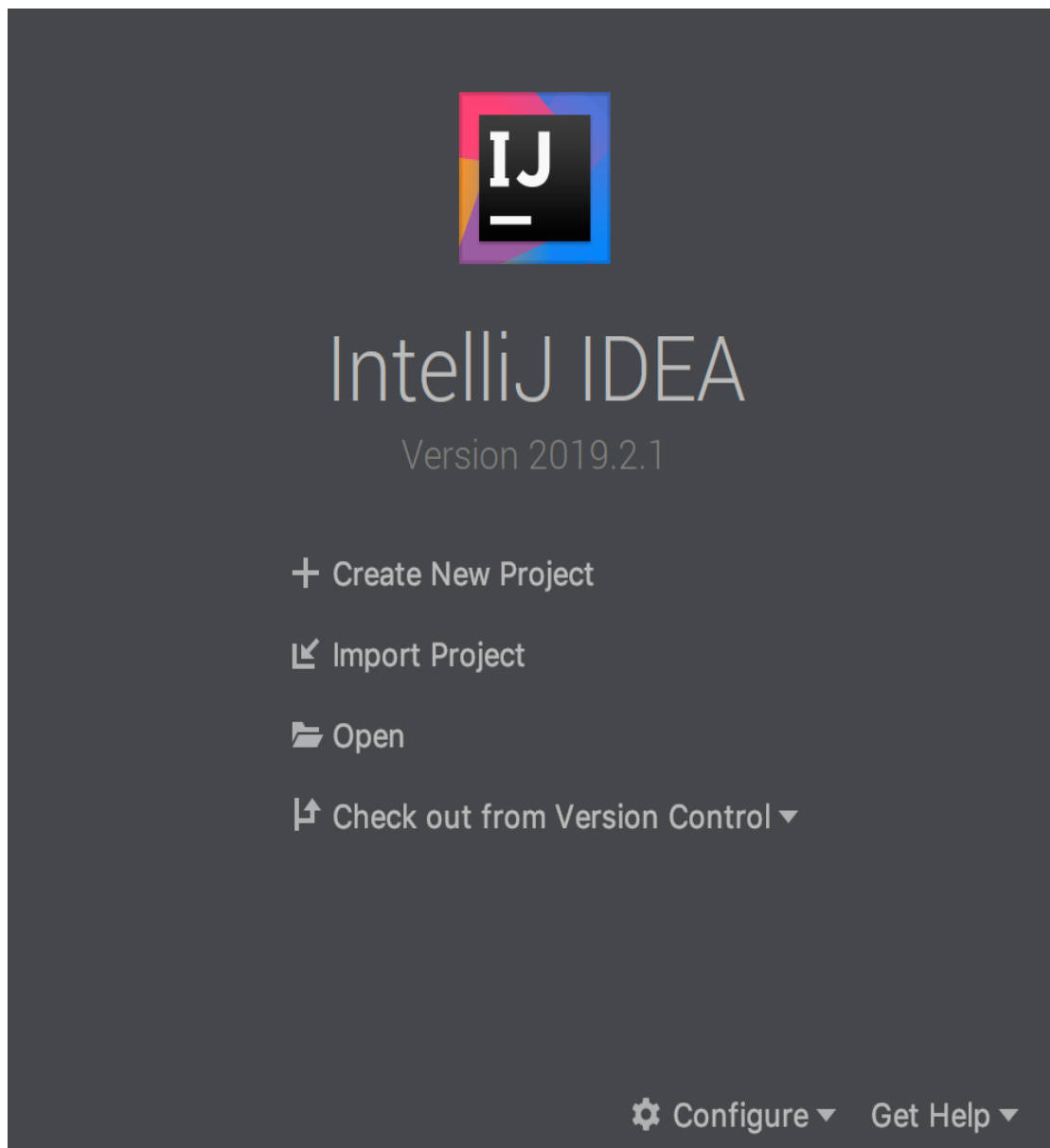


这个直接用浏览器进行下载，可能速度是有点慢的，所以建议大家在浏览器下载界面里选择拷贝链接，然后打开迅雷之类的工具去下载，速度会非常的快。

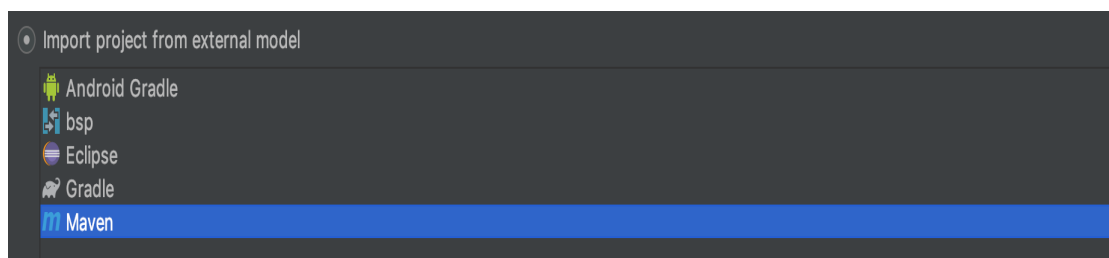
接着在你本地会发现一个master.zip包，解压缩之后是rocketmq-master目录，这里就存放了RocketMQ最新的master分支的源码了。

## 2、将RocketMQ源码导入到IntelliJ IDEA里去

接着我们需要将RocketMQ源码导入到IntelliJ IDEA里去，大家打开IntelliJ IDEA比较新的版本的启动界面，会看到下面的界面：



接着点击上图中的Import Project，就是导入一个项目，因为RocketMQ源码本身下载下来就是一个已有的项目了，所以直接导入就可以了，此时会看到一个本地目录选择框，你就选择你的rocketmq-master目录就可以了，接着进入下面的界面。



在这个界面里，你就选择Import project from external model，然后选择里面的Maven就可以了，因为RocketMQ是基于maven来进行构建的，接着会进入下面的界面。

Project format: .idea (directory based) ▾

☐ Keep project files in:

☐ Import Maven projects automatically

☒ Detect compiler automatically

☒ Create IntelliJ IDEA modules for aggregator projects (with 'pom' packaging)

☐ Create module groups for multi-module Maven projects

☒ Keep source and test folders on reimport

☒ Exclude build directory (%PROJECT\_ROOT%/target)

☒ Use Maven output directories

Generated sources folders: Detect automatically ▾

Phase to be used for folders update: process-resources ▾

IDEA needs to execute one of the listed phases in order to discover all source folders that are configured via Maven plugins.  
Note that all test-\* phases firstly generate and compile production sources.

Automatically download: ☐ Sources ☐ Documentation

Dependency types: jar, test-jar, maven-plugin, ejb, ejb-client, jboss-har, jboss-sar, war, ear, bundle

Comma separated list of dependency types that should be imported

Help Cancel Previous Next

这里你什么都不用管，你就直接点击Next就可以了，后续会出现一系列的界面，你都直接点击Next、Next就可以了，下面我给出你接下来会看到的一些界面的截图。

Select profiles:

☐ apache-release

☐ it-test

☐ sonar-apache

☒ jdk8

☐ release-sign-artifacts

Help Cancel Previous Next

Select Maven projects to import

☒ org.apache.rocketmq:rocketmq-all:4.6.0

Select all

Unselect all

☐ Open Project Structure after import

Help

Cancel

Previous

Next

+ -

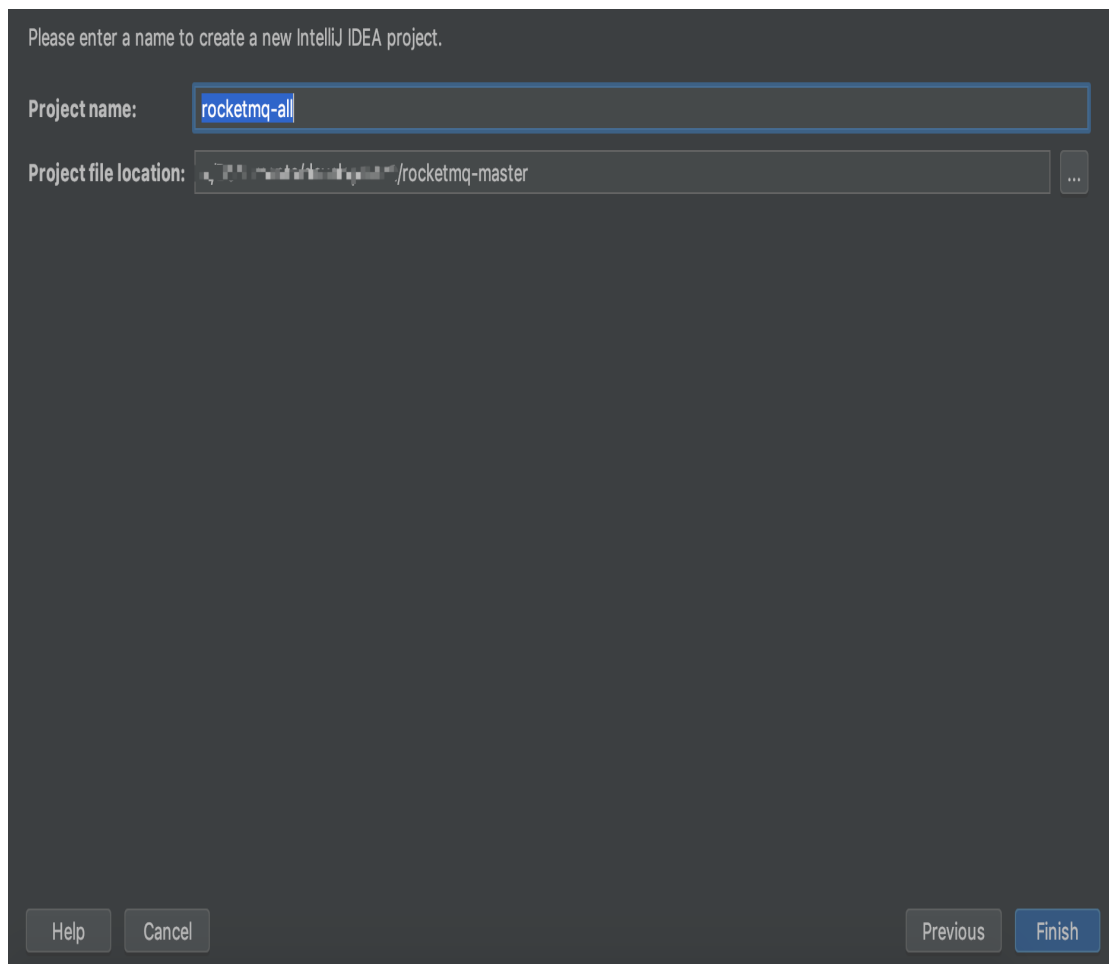
1.8

Name: 1.8

JDK home path: /Library/Java/JavaVirtualMachines/1.8.0\_102.jdk/Contents/Home

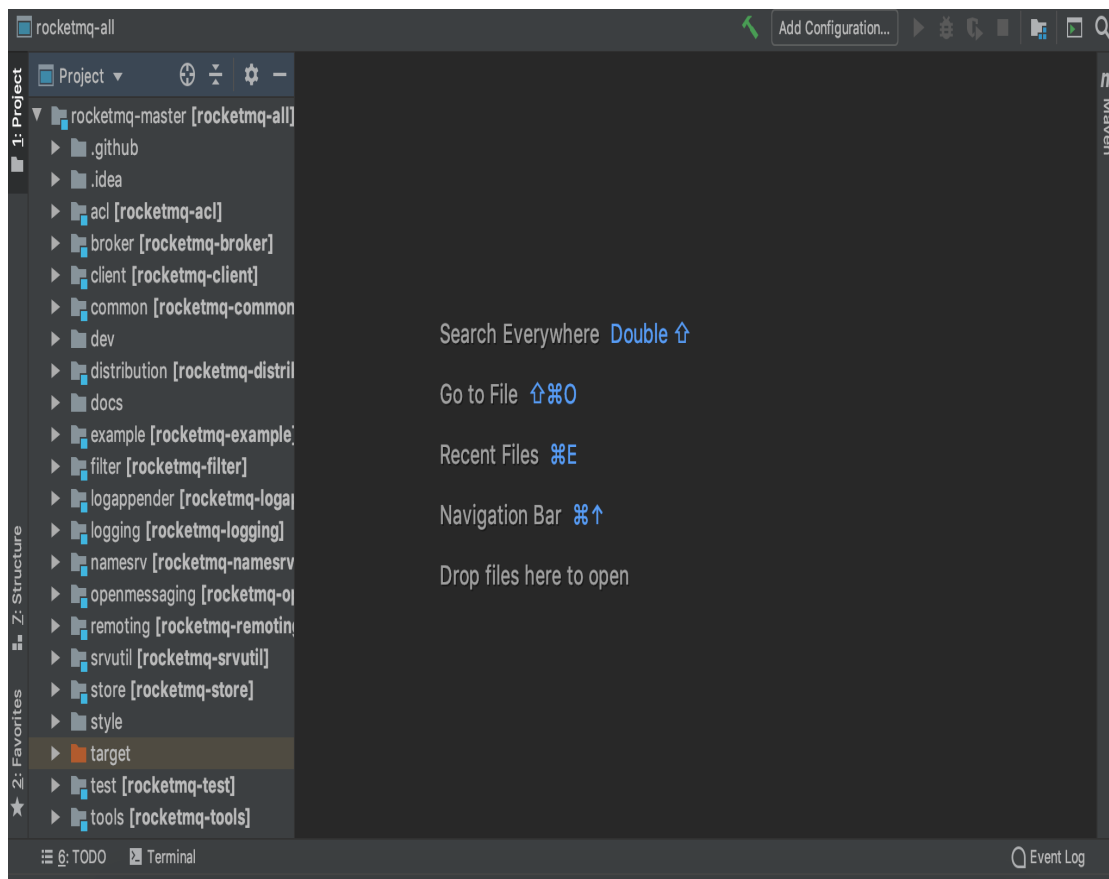
Classpath Sourcepath Annotations Document Path

Library: /Library/Java/JavaVirtualMachines/1.8.0\_102.jdk/Contents/Home/lib/charsets.jar  
Library: /Library/Java/JavaVirtualMachines/1.8.0\_102.jdk/Contents/Home/lib/deploy.jar  
Library: /Library/Java/JavaVirtualMachines/1.8.0\_102.jdk/Contents/Home/lib/ext/class.jar  
Library: /Library/Java/JavaVirtualMachines/1.8.0\_102.jdk/Contents/Home/lib/ext/class\_loader.jar  
Library: /Library/Java/JavaVirtualMachines/1.8.0\_102.jdk/Contents/Home/lib/ext/rt.jar  
Library: /Library/Java/JavaVirtualMachines/1.8.0\_102.jdk/Contents/Home/lib/ext/sunec.jar  
Library: /Library/Java/JavaVirtualMachines/1.8.0\_102.jdk/Contents/Home/lib/ext/sunec\_provider.jar  
Library: /Library/Java/JavaVirtualMachines/1.8.0\_102.jdk/Contents/Home/lib/ext/sunpkcs11.jar  
Library: /Library/Java/JavaVirtualMachines/1.8.0\_102.jdk/Contents/Home/lib/ext/zipfs.jar  
Library: /Library/Java/JavaVirtualMachines/1.8.0\_102.jdk/Contents/Home/lib/javaws.jar  
Library: /Library/Java/JavaVirtualMachines/1.8.0\_102.jdk/Contents/Home/lib/jce.jar  
Library: /Library/Java/JavaVirtualMachines/1.8.0\_102.jdk/Contents/Home/lib/jfr.jar  
Library: /Library/Java/JavaVirtualMachines/1.8.0\_102.jdk/Contents/Home/lib/jfxswt.jar  
Library: /Library/Java/JavaVirtualMachines/1.8.0\_102.jdk/Contents/Home/lib/jfxswt.jar



从上图中可以看到我们拉取的是RocketMQ最新的稳定版本，4.6.0的版本的源码，一直到最后一个图，大家就直接点击Finish按钮就可以了，这就初步完成了源码的导入，接着会进入到IntelliJ IDEA的项目界面里去。

这个时候，你会看到有一个进度条会显示他在加载RocketMQ源码的所有依赖包的进度，然后你需要等待很长时间，根据每个人网速的不同，可能需要等待几十分钟到几个小时都有可能，接着你会看到如下所示的项目界面。



但是你之前下载依赖包的过程可能会有一些失败的情况，比如因为网络超时导致jar包没下载下来，此时你可以进入命令行，无论是windows或者是mac都可以进入命令行的，然后进入到rocketmq-master项目的目录中，接着执行mvn clean install。

此时就是说用maven对项目执行一下清理、编译和部署到本地仓库，接着maven会自动下载一些之前下载失败的依赖包，然后你会看到maven的BUILD SUCCESS的提示，就说明这个项目彻底弄好了。

然后我们就搞定了RocketMQ的源码项目了，这个项目就导入到IntelliJ IDEA里去了。

### 3、RocketMQ的源码目录结构

接着我们简单给大家说一下RocketMQ的源码目录结构：

**broker**：顾名思义，这个里面存放的就是RocketMQ的Broker相关的代码，这里的代码可以用来启动Broker进程

**client**：顾名思义，这个里面就是RocketMQ的Producer、Consumer这些客户端的代码，生产消息、消费消息的代码都在里面

**common**：这里放的是一些公共的代码

**dev**：这里放的是开发相关的一些信息

**distribution**：这里放的就是用来部署RocketMQ的一些东西，比如bin目录，conf目录，等等

**example**：这里放的是RocketMQ的一些例子

**filter**：这里放的是RocketMQ的一些过滤器的东西

**logappender和logging**：这里放的是RocketMQ的日志打印相关的东西

**namesrv**：这里放的就是NameServer的源码

**openmessaging**：这是开放消息标准，这个可以先忽略

**remoting**：这个很重要，这里放的是RocketMQ的远程网络通信模块的代码，基于netty实现的

**srvutil**：这里放的是一些工具类

**store**：这个也很重要，这里放的是消息在Broker上进行存储相关的一些源码

**style、test、tools**：这里放的是checkstyle代码检查的东西，一些测试相关的类，还有就是tools里放的一些命令行监控工具类

### 4、边干边学

希望大家看完这篇文章就可以跟着把RocketMQ的源码下载下来，然后导入到IntelliJ IDEA里去，然后先看一下他整体的源码结构，跟着动手做一下，下一篇文章我们来讲解如何在IntelliJ IDEA中启动RocketMQ来本地调试他的源码。